

CONCURRENT MODELING & SIMULATION IN JAVA

I.C. Legrand

CATEGORIES OF SIMULATION MODELS

Continuous time -> usually solved by sets of differential equations.

Discrete time -> Systems which are considered only at selected moments in time.

Continuous time- discrete event

Discrete event simulation: series of interacting events which require scheduling based on an internal “clock”

EVENT ORIENTED

PROCESS ORIENTED

Process oriented DES Based on “ACTIVE OBJECTS”

thread: execution of a piece of code that occurs independently of and possibly concurrently with another one.

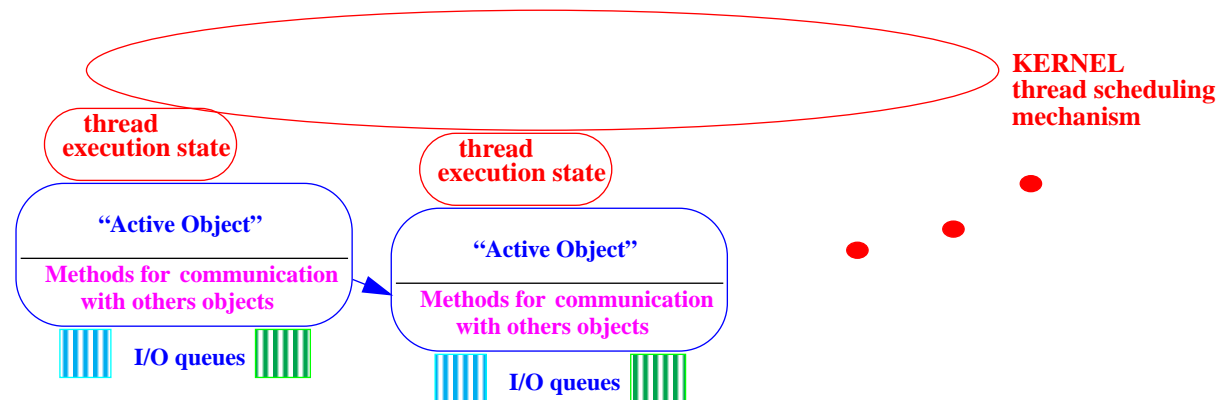
may be: blocked (waiting for some event) running ready

execution state set of state information needed to permit concurrent execution

may be: active / inactive

mutual exclusion: mechanism that allow an action to performed on an object without interruption

asynchronous interaction: signals/ semaphores for interrupting



SIMULATION MODEL

The simulation/modeling task for the MONARC project requires to describe complex programs running in a distributed environment.

Necessary to abstract from the real system all components and their interaction.

THE MODEL has to be equivalent to the simulated system in all important respects.

Selecting Tools which allows easily to map the logical model into the simulation environment.

For such type of applications, the simulation tool(s) should provide basic support for **executable entities**. In most uses, these executable entities (“active objects” or “actors”) conceptually (if not actually) execute concurrently. Such active objects extend the concept of objects to concurrent computation by encapsulating a thread of control and having interfaces to interact with other objects.

This process oriented approach for discrete event simulation is well suited to **describe concurrent running programs**. The “active objects” (having an execution thread, a program counter, stack...) allow an easy way to map the structure of a complex

Java has build in **multi-thread** support for concurrent processing. Usually the implementation is done at the Operating System (lightweight processes). The Threads class in Java is for concurrent programming not simulation. However, simulation packages can be developed by using the multi-thread structure and providing a new scheduler to be used with the threaded objects for discrete event simulations.

Packages which also provide support for **distributed objects** (through RMI or CORBA) can be used on distributed simulations, or for an environment in which parts of the system are simulated and interfaced through such a mechanism with other parts which actually are running the real application. The distributed object model can also provide the environment to be used for autonomous mobile agents.

A few discrete event simulation packages based on a multi-thread process oriented design developed in JAVA

PTOMEMY II (University of California at Berkeley)

Heterogeneous Concurrent Modelling and Design in JAVA.

Complete new redesign of the Ptolomy simulation package. It can provide a framework for “open distributed object-oriented systems”.

Ptolomy II is a set of Java packages supporting heterogeneous, concurrent modelling and design. Has distributed object support.

Parsimony (University of Waterloo)

Parsimony stands for PARallel SIMulator Once Named Yaddes. And Yaddes stands for Yet Another Distributed Discrete Event Simulator.

The Parsimony Project is a vehicle for conducting research in distributed, network-centric computing. The main objective is the development of a Java-based testbed for distributed interactive simulation. Has a distributed object support and graphics.

Silk (ThreadTec, Inc.)

Silk is a Java-based modelling tool for the simulation, study and improvement of industrial systems.

Silk is a complete package of process-oriented simulation constructs including JavaBeans components for visual modelling. Includes a variety of methods for entity generation, resource scheduling and output post-processing using Microsoft Execl.

Simjava (The University of Edinburgh)

Simjava is Java Based version of Hierarchical computer Architecture design and Simulation Environment (HASE).

Simjava is a process based discrete event simulation package for Java, similar to Jade's Sim++, with animation facilities.

Distributed simjava is an extension of simjava based on RMI.

JSIM (University of Georgia)

JSIM is a Java-based simulation and animation environment supporting Web-Based Simulation.

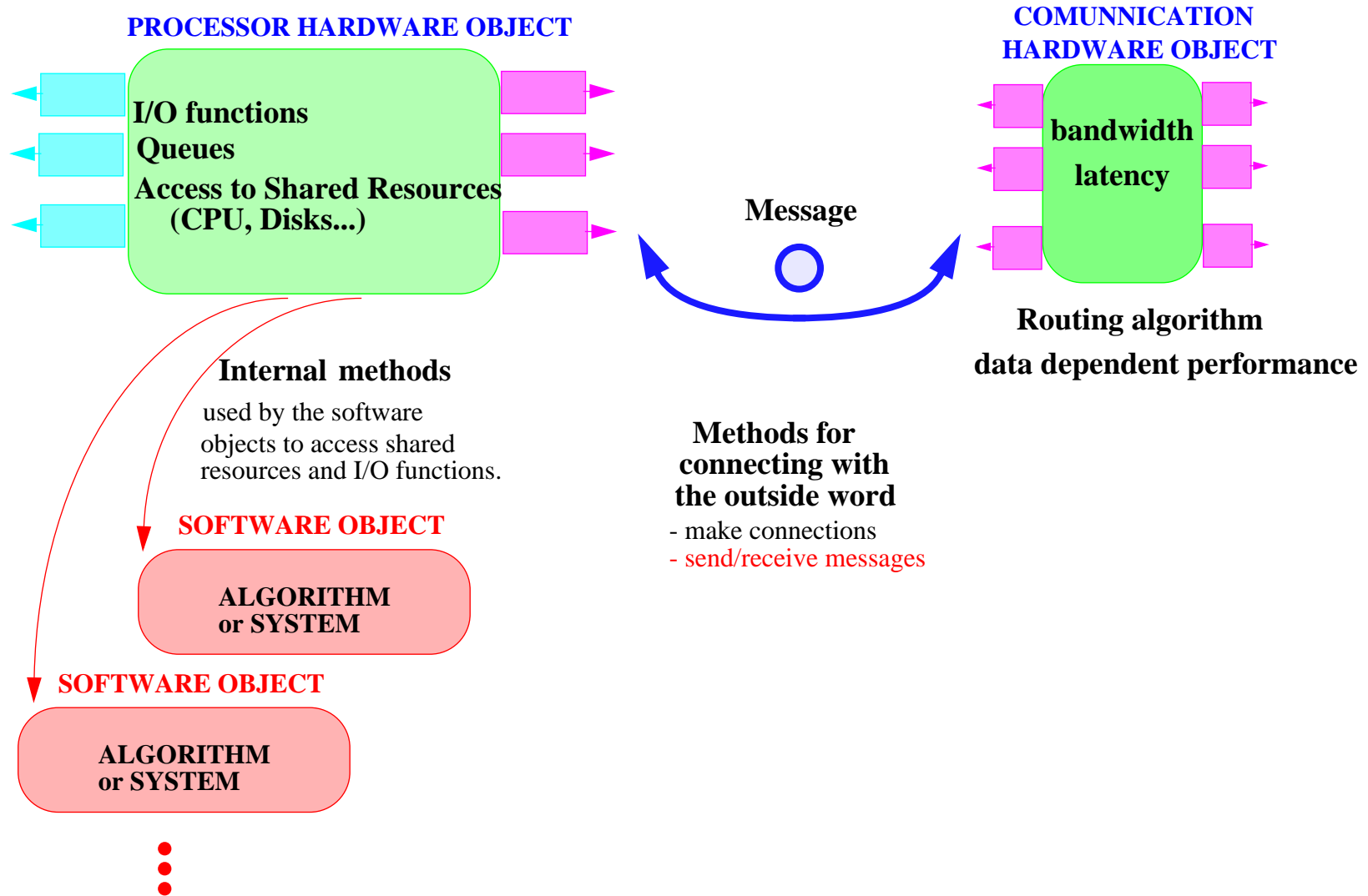
Simulation models may be built using either the event package (Event-Scheduling Paradigm) or the process package (Process-Interaction Paradigm).

In addition, a graphical designer (model package) allows process models to be rapidly built graphically.

JavaSim (Department of Computing Science, University of Newcastle upon Tyne)

JavaSim is a set of Java packages for building discrete event process-based simulation, similar to that in Simula and C++SIM (from which JavaSim is derived).

BASIC OBJECTS FOR SIMULATING A DISTRIBUTED PROCESSING SYSTEM



CS2 system simulation

Applet Viewer: CS2.class

Applet

Nr. of Senders	4	Disk Servers	4
<DAQ Message Size> [MB]:	5	Distributed	Neg. Exponential
<Sender's period> [s]:	1	Distributed	Normal SD=10%
<Event Size> [MB]:	2	Distributed	Normal SD=30%
<Processing time> [s]:	2	Distributed	Normal SD=40%

Show Messages Show Time Diagrams:

Nodes connected to DAQ

The diagram illustrates a network topology. At the top, four nodes labeled 'S' and 'R' are connected to a central network. Below this, a central box contains the text 'Total Bandwidth : 30.0'. To the right, four 'Disk Server Nodes' are shown, each with a 'W' icon. At the bottom, a row of nodes labeled 'W' and 'R' represents 'Data Processing Nodes'. The network is represented by a central area with a colorful, textured background.

Running: sim time = 54.1727

Layout Run Restart Stop

Speed: 10

Applet started.

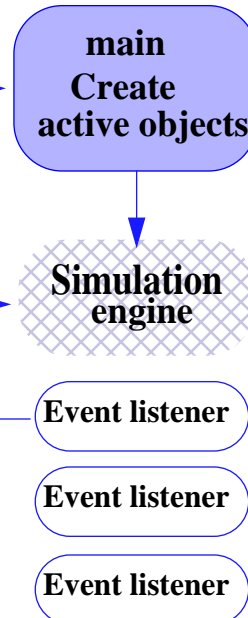
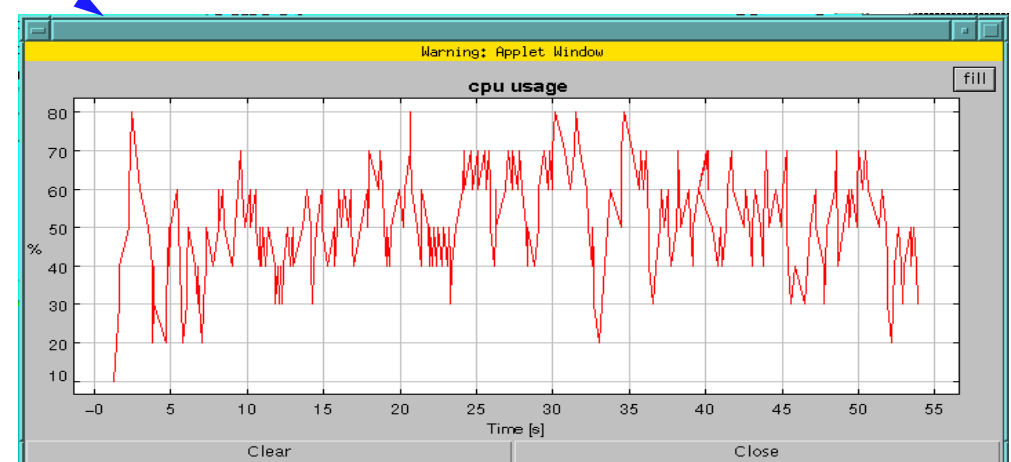
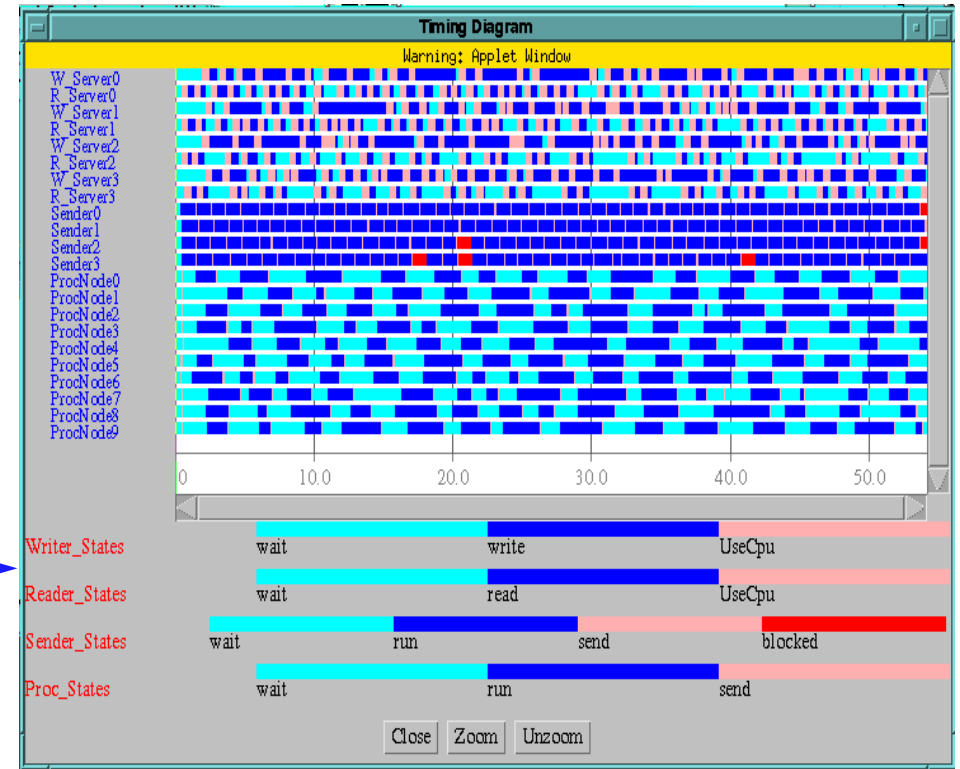
appletviewer <http://home.cern.ch/~cil/cs.html>

CS2 Simulation - Graphic Tools

GUI



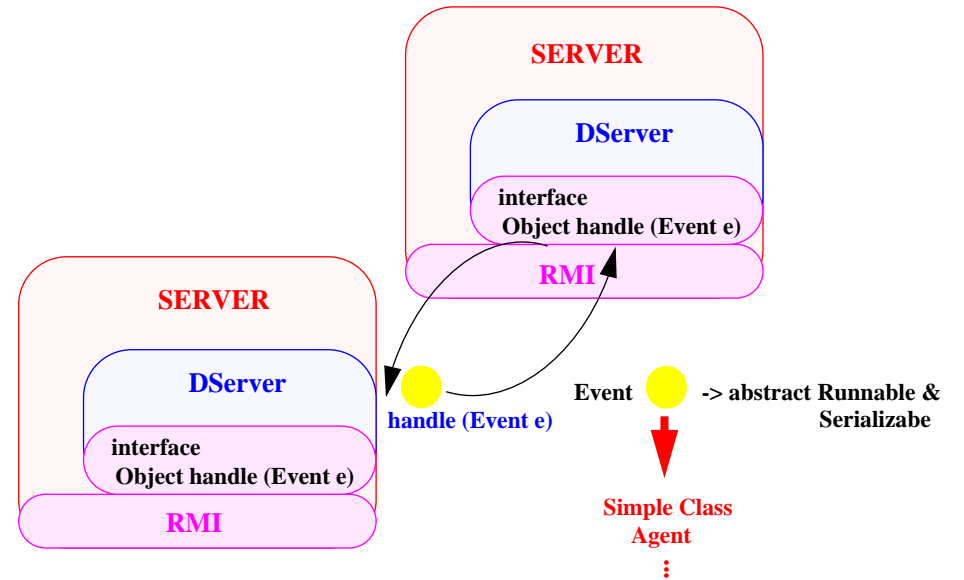
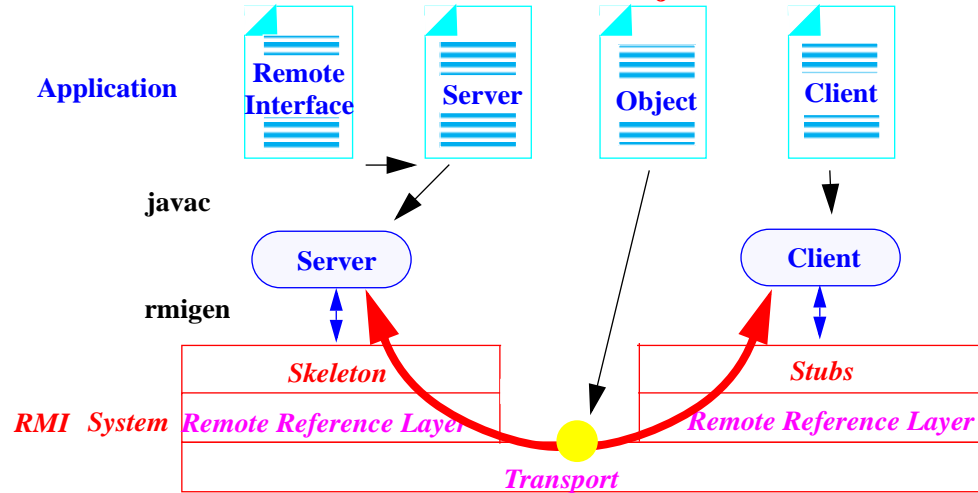
“Logic Analyzer”



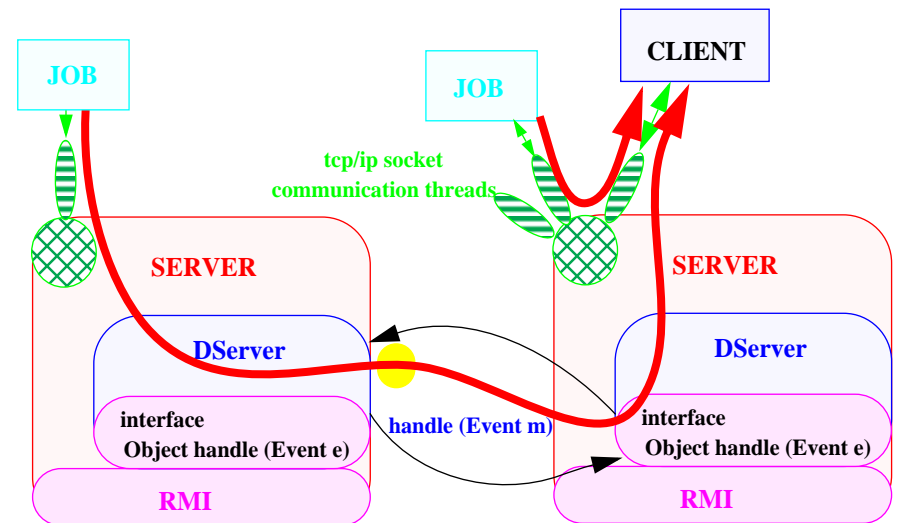
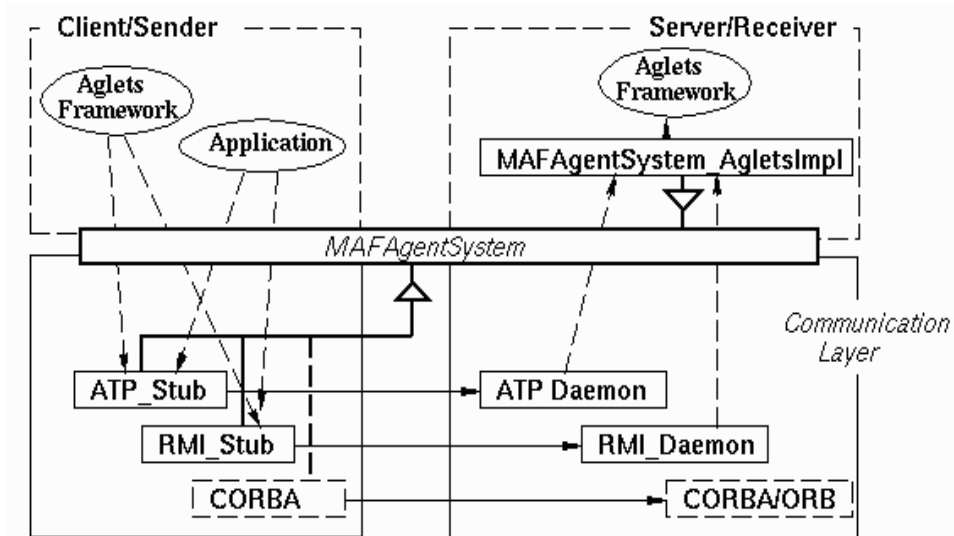
Java Beans

DISTRIBUTED OBJECT SYSTEMS

Remote Method Invocation & Object Serialization



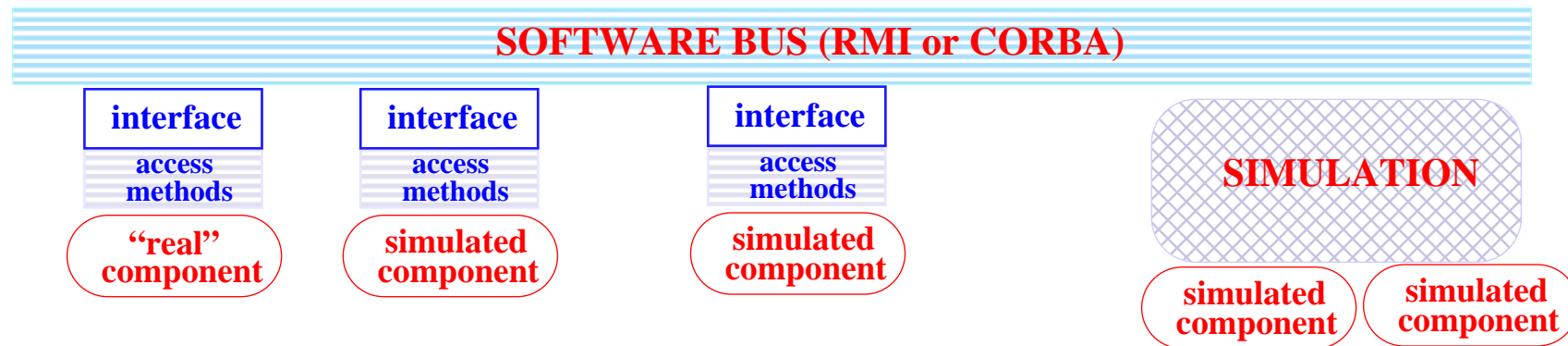
Mobile Agents



DISTRIBUTED SIMUALTION

Components of the simulated system may be implemented on remote servers via RMI (or CORBA). They must implement an Interface which is exported through such a distributed object model.

- * Components may be parametrized where the values are supplied by a remote server
- * Reduced-order modelling scheme where behaviour of subsystems is provided by independent servers.
- * Use “real-application” (which implement the same interface) parts in the simulation program



SUMMARY

Define the LOGICAL MODEL for system.

Use a tool which provides an easy way to map the LOGICAL MODEL.

Graphic tools are necessary.

The Java programming environment provides the right tools for developing a flexible and distributed process oriented simulation.

Such an approach for simulation:

allows an easy way to map the distributed data processing task into the simulation

can be distributed

may interact with real components

can handle dynamically any system configuration

can be quite easily interfaced with different graphics tools